# Generic Programming

CSC 209 Data Structures

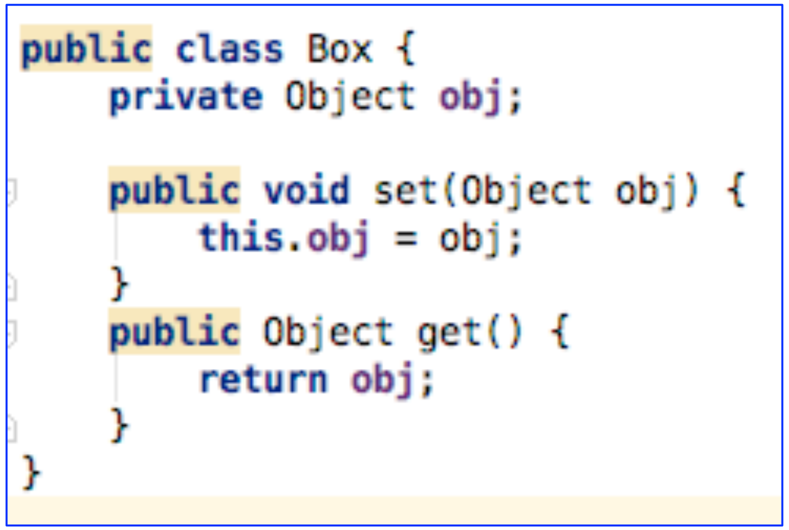
**Kulwadee Somboonviwat**

kulwadee.som [at] sit.kmutt.ac.th

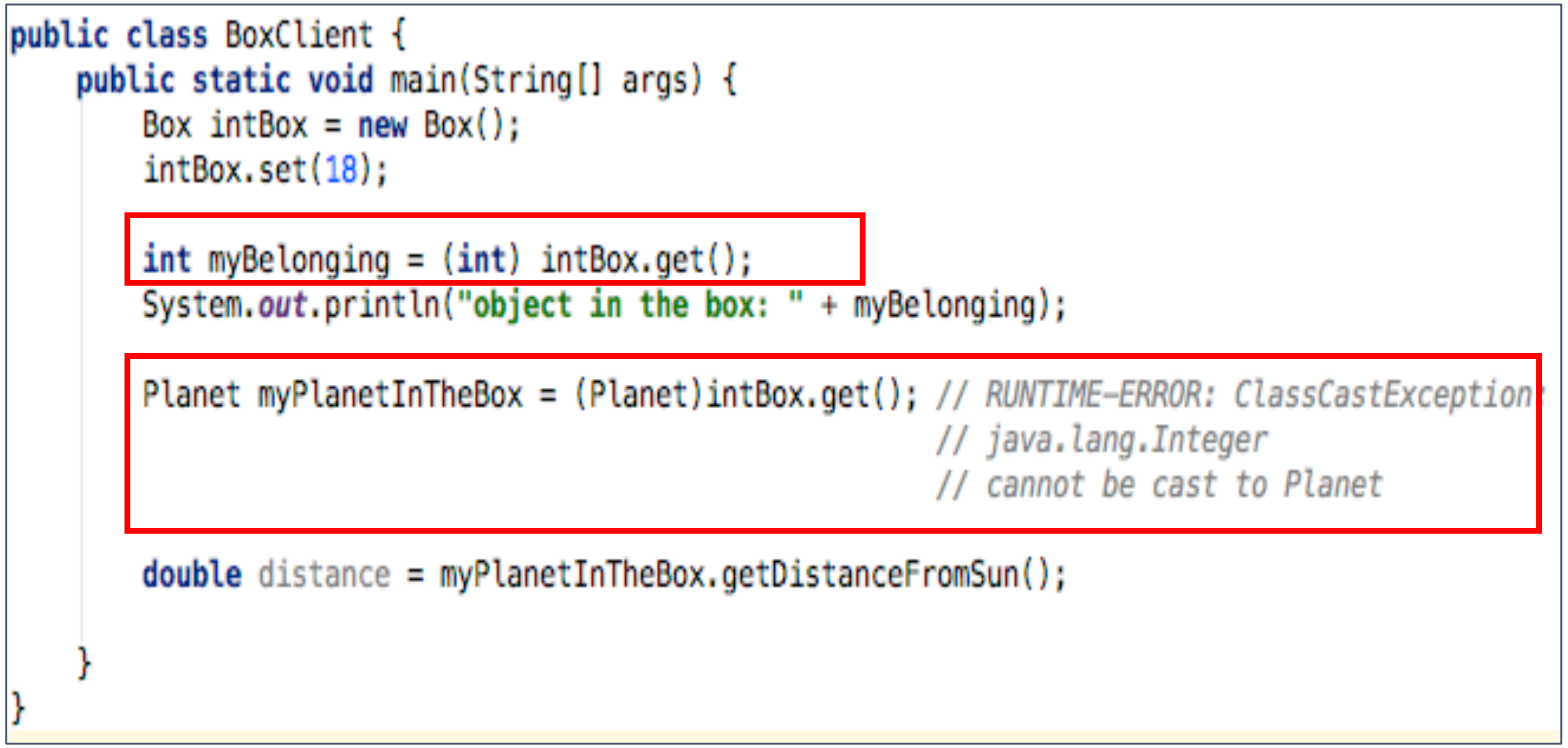
## Generic Programming

- Writing code that can be reused for objects of many different types

- Why generic programming?
  - **Disadvantages of non-generic code**: Casting and Lack of compile-time error checking

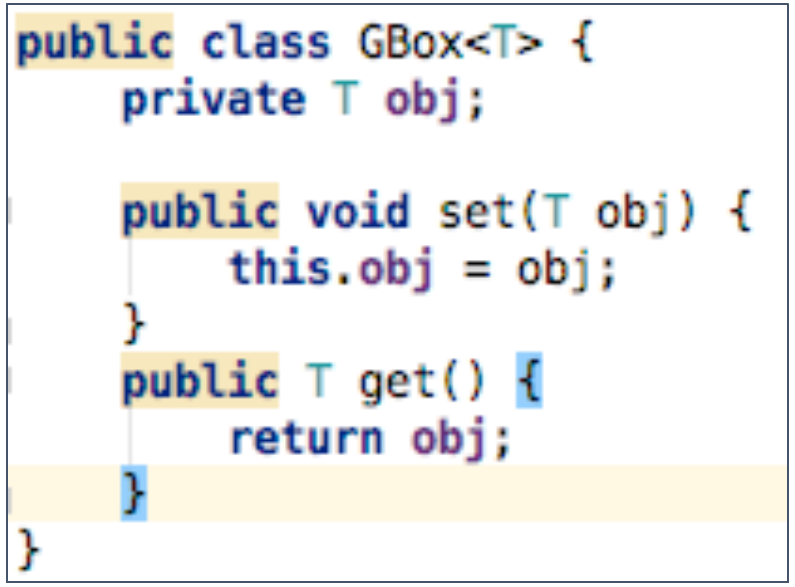


```java
public class Box {
    private Object obj;

    public void set(Object obj) {
        this.obj = obj;
    }
    public Object get() {
        return obj;
    }
}
```

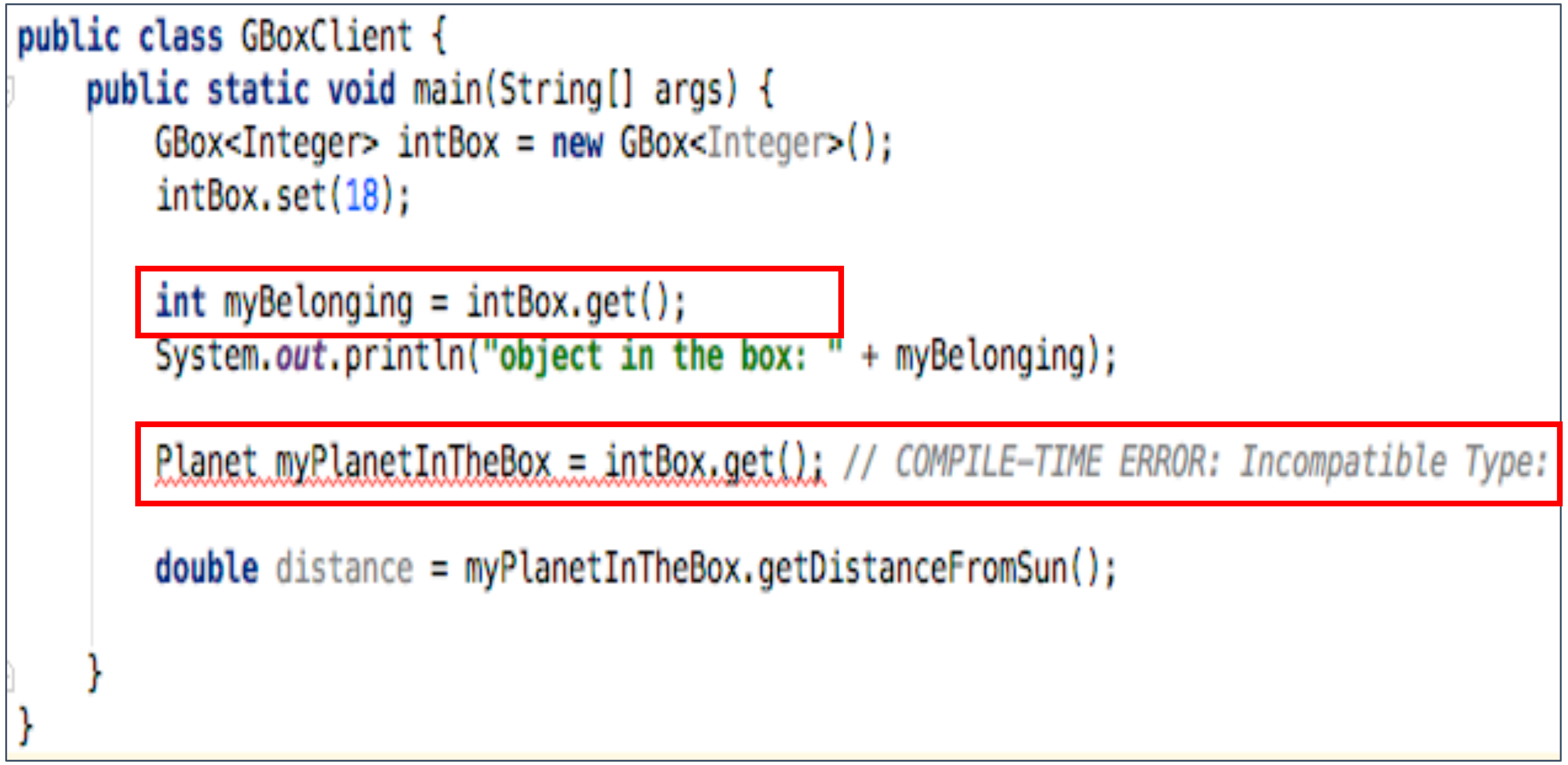


```java
public class BoxClient {
    public static void main(String[] args) {
        Box intBox = new Box();
        intBox.set(18);

        int myBelonging = (int) intBox.get();
        System.out.println("object in the box: " + myBelonging);

        Planet myPlanetInTheBox = (Planet)intBox.get(); // RUNTIME-ERROR: ClassCastException
                                                        // java.lang.Integer
                                                        // cannot be cast to Planet

        double distance = myPlanetInTheBox.getDistanceFromSun();

    }
}
```

# Box Class (Generic version): no casting, compile-time error

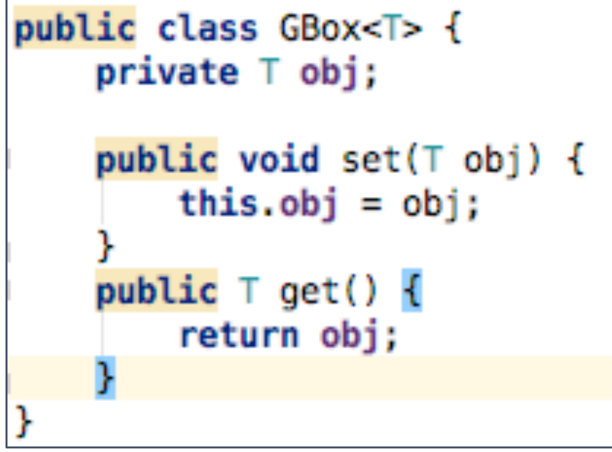


```java
public class GBox<T> {
    private T obj;

    public void set(T obj) {
        this.obj = obj;
    }
    public T get() {
        return obj;
    }
}
```



```java
public class GBoxClient {
    public static void main(String[] args) {
        GBox<Integer> intBox = new GBox<Integer>();
        intBox.set(18);

        int myBelonging = intBox.get();
        System.out.println("object in the box: " + myBelonging);

        Planet myPlanetInTheBox = intBox.get(); // COMPILE-TIME ERROR: Incompatible Type:

        double distance = myPlanetInTheBox.getDistanceFromSun();

    }
}
```

# Defining a simple generic class



```
public class GBox<T> {
    private T obj;

    public void set(T obj) {
        this.obj = obj;
    }
    public T get() {
        return obj;
    }
}
```

- The **T** in public class Box<T> is a ***type variable***.
- The type variable is used throughout the class definition.

  private T obj;

- When instantiating an object of a generic class, you must provide the actual type of the type variable.

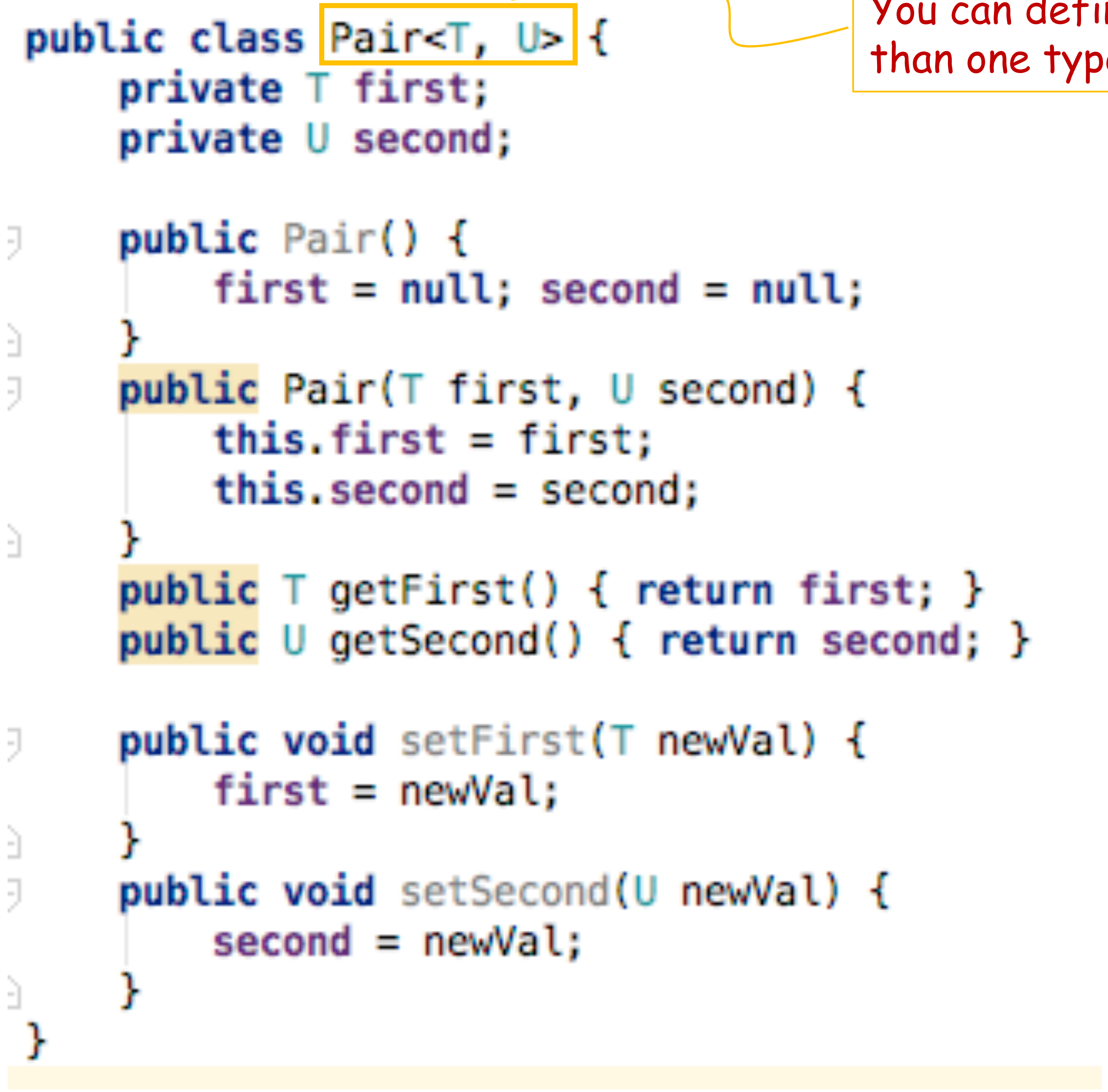
  GBox<String> myBox = new GBox<String>();

- Now, you can think of myBox as an object with the following methods:

  private String obj;

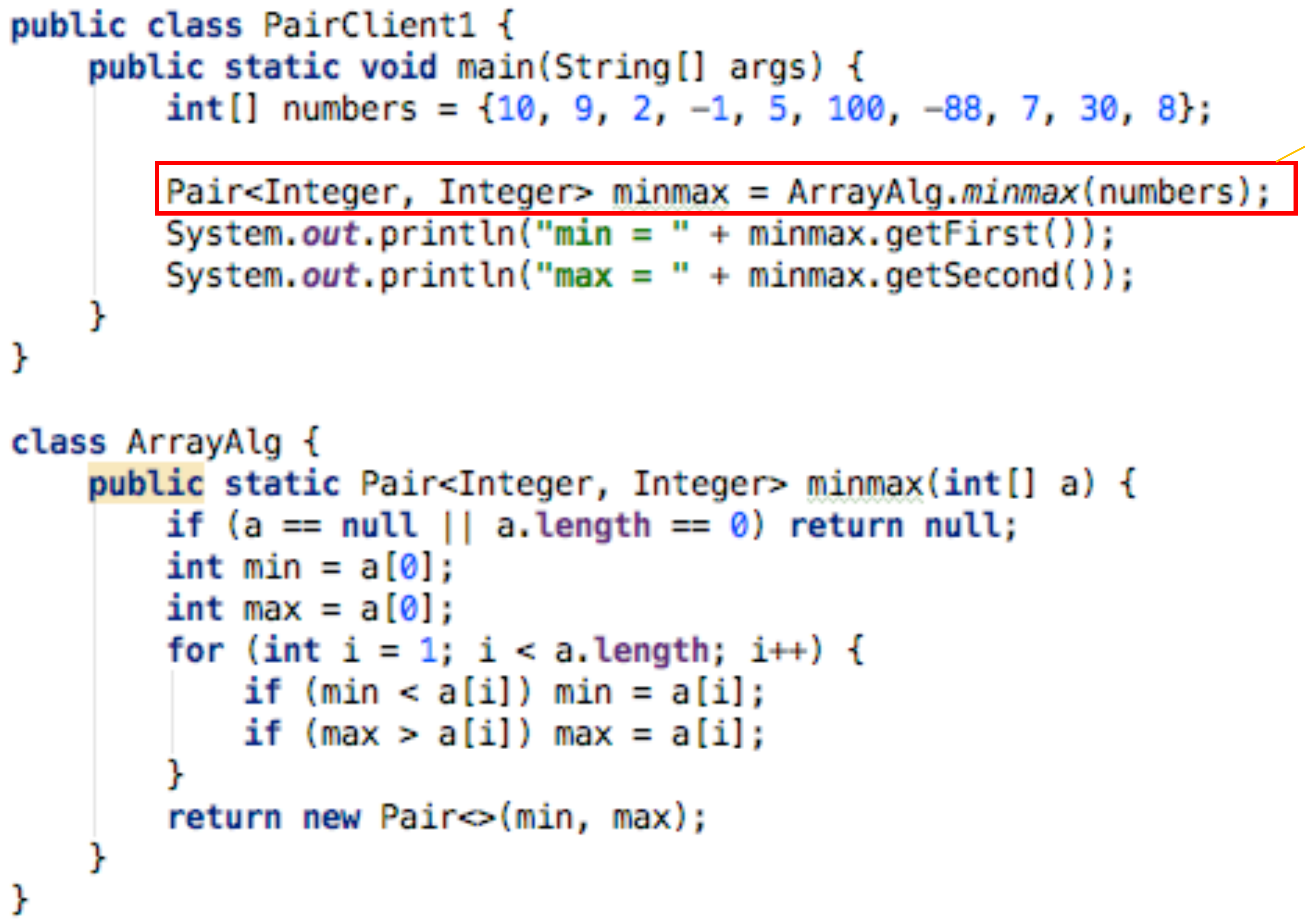
  public void set(String obj) { … }

  public String get() { … }



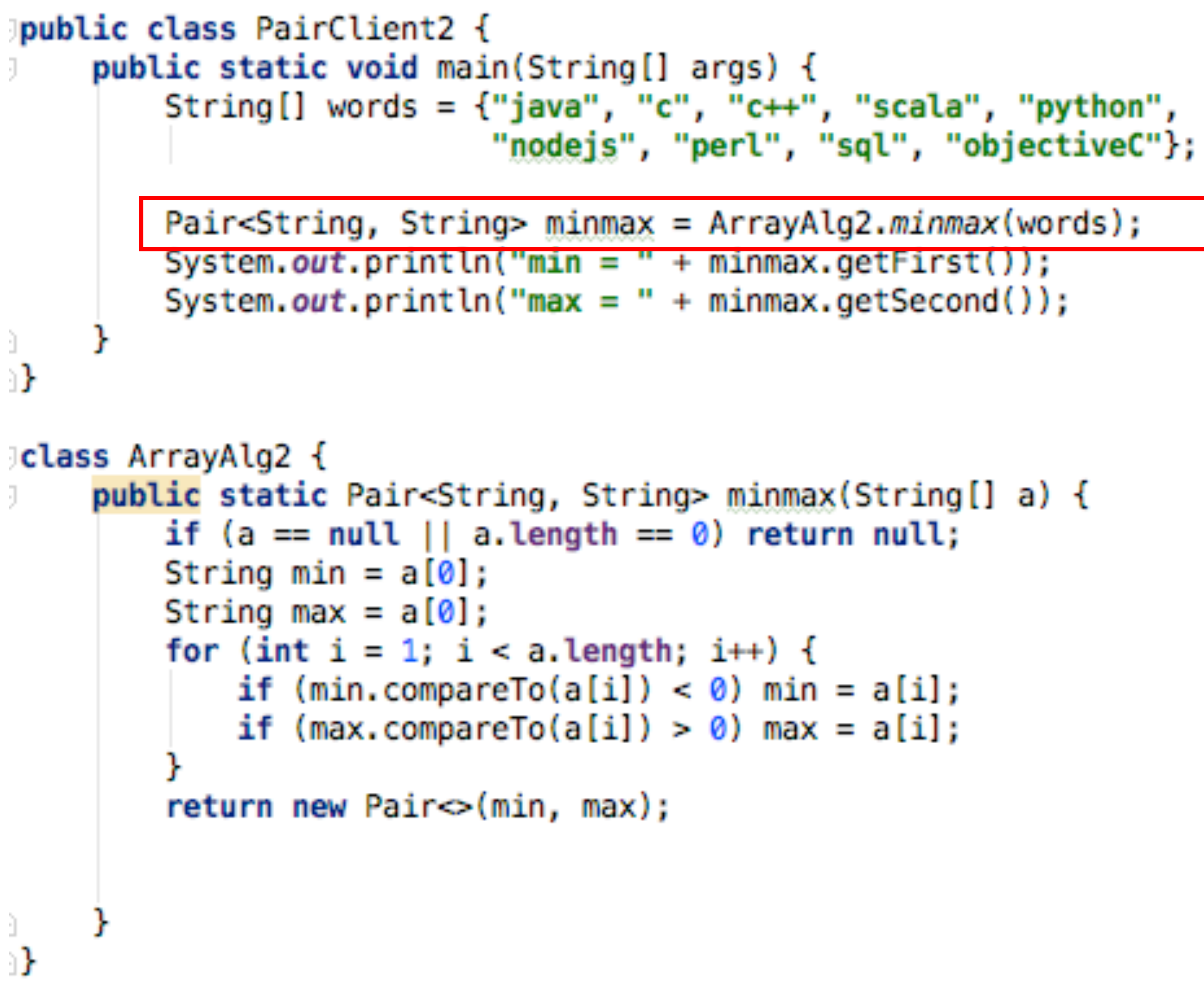
```java
public class Pair<T, U> {
    private T first;
    private U second;

    public Pair() {
        first = null; second = null;
    }
    public Pair(T first, U second) {
        this.first = first;
        this.second = second;
    }
    public T getFirst() { return first; }
    public U getSecond() { return second; }

    public void setFirst(T newVal) {
        first = newVal;
    }
    public void setSecond(U newVal) {
        second = newVal;
    }
}
```

You can define more than one type variable



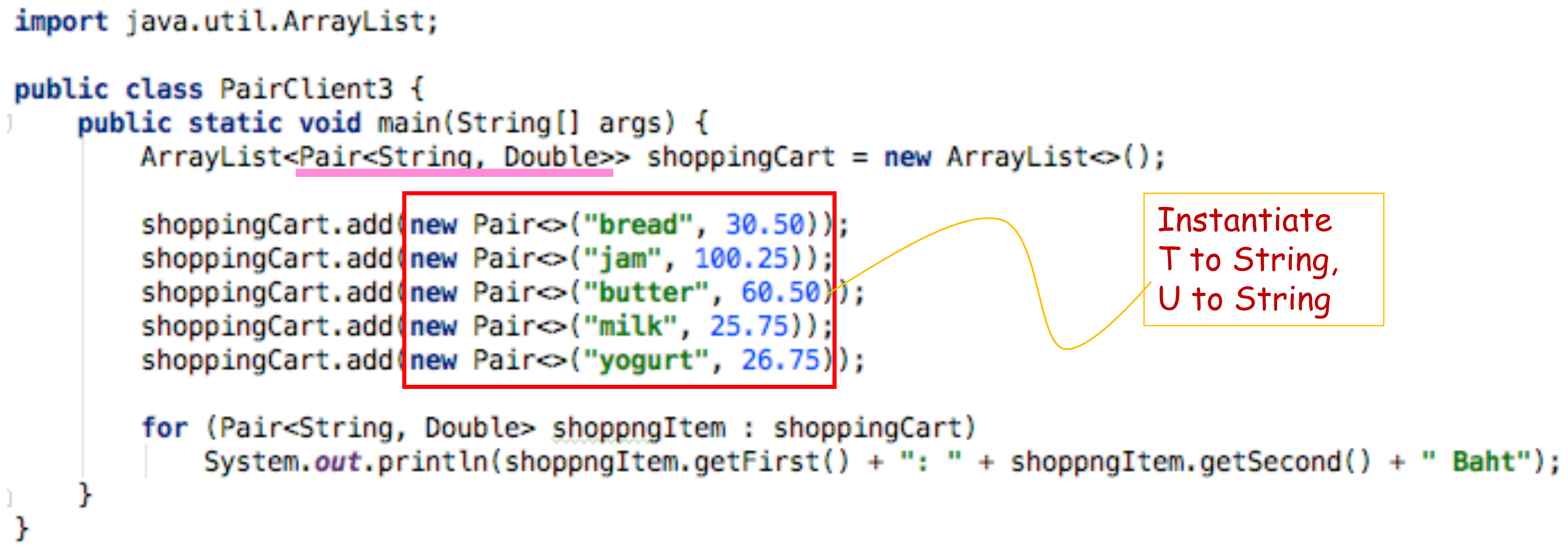
```java
public class PairClient1 {
    public static void main(String[] args) {
        int[] numbers = {10, 9, 2, -1, 5, 100, -88, 7, 30, 8};

        Pair<Integer, Integer> minmax = ArrayAlg.minmax(numbers);
        System.out.println("min = " + minmax.getFirst());
        System.out.println("max = " + minmax.getSecond());
    }
}

class ArrayAlg {
    public static Pair<Integer, Integer> minmax(int[] a) {
        if (a == null || a.length == 0) return null;
        int min = a[0];
        int max = a[0];
        for (int i = 1; i < a.length; i++) {
            if (min < a[i]) min = a[i];
            if (max > a[i]) max = a[i];
        }
        return new Pair<>(min, max);
    }
}
```

Instantiate T to Integer, U to Integer



```java
public class PairClient2 {
    public static void main(String[] args) {
        String[] words = {"java", "c", "c++", "scala", "python",
                          "nodejs", "perl", "sql", "objectiveC"};

        Pair<String, String> minmax = ArrayAlg2.minmax(words);
        System.out.println("min = " + minmax.getFirst());
        System.out.println("max = " + minmax.getSecond());
    }
}

class ArrayAlg2 {
    public static Pair<String, String> minmax(String[] a) {
        if (a == null || a.length == 0) return null;
        String min = a[0];
        String max = a[0];
        for (int i = 1; i < a.length; i++) {
            if (min.compareTo(a[i]) < 0) min = a[i];
            if (max.compareTo(a[i]) > 0) max = a[i];
        }
        return new Pair<>(min, max);


    }
}
```

Instantiate
T to String,
U to String



```java
import java.util.ArrayList;

public class PairClient3 {
    public static void main(String[] args) {
        ArrayList<Pair<String, Double>> shoppingCart = new ArrayList<>();

        shoppingCart.add(new Pair<>("bread", 30.50));
        shoppingCart.add(new Pair<>("jam", 100.25));
        shoppingCart.add(new Pair<>("butter", 60.50));
        shoppingCart.add(new Pair<>("milk", 25.75));
        shoppingCart.add(new Pair<>("yogurt", 26.75));

        for (Pair<String, Double> shoppngItem : shoppingCart)
            System.out.println(shoppngItem.getFirst() + ": " + shoppngItem.getSecond() + " Baht");
    }
}
```

Instantiate
T to String,
U to String